# Software Requirements Specification

**for**

# POWERstation Kiosk

Version 1.0

June 3, 2008

**Prepared by:** Andrew S. Kaplan

## Table of Contents

# 1. Introduction

## A. Purpose

This Software Requirements Specification (SRS) is intended to provide a detailed description of the use case and functionality of the POWERstation Kiosk (Kiosk) system. This document will outline the major features intended for inclusion in the Kiosk system (as well as minor desired features) and will offer a preliminary glimpse of the User Interface (UI). The document will also cover hardware, software, and various other technical dependencies.

## B. Definitions

This document features some terminology which readers may be unfamiliar with. See Appendix A (Glossary) for a list of these terms and their definitions.

## C. Target Audience

This document is intended for all individuals participating in and/or supervising the Kiosk project. Less technical readers will want to concentrate on the remainder of this Introduction section and the next section (Overall Description). Also of some interest to these readers may be Section 3.f where we cover Reliability, Availability, Security, and Maintainability.

Section 3 (Specific Requirements) contains more detailed technical requirements and will be of greater interest to the more technically-oriented readers. It includes information on external interfaces and hardware.

## D. Project Scope

The Kiosk system is Windows-based software designed to run on commodity hardware to provider a patient-friendly application for various functions that enable self-service appointment check-in.

# 2. Overall Description

## 2.1 Product Perspective

The Kiosk project is intended to enable patients to check themselves in for their appointments and perform various tasks necessary to facility the check-in process and its ancillary requirements such as form signing and co-payment collection.

The Kiosk is intended to run on Windows either directly on the Kiosk hardware or on a Windows server and accessed via Remote Desktop Protocol, the latter providing easier central management, security, and disaster recovery.

There is also a server-side component of the Kiosk that includes the database, data synchronization services, and the Kiosk dashboard components. Also, when utilizing thin-client mode (RDP), the server hosts the client application software itself and runs it within its own memory space using its own processor. The Kiosk hardware will run only a small amount of software known as the RDP client software. Microsoft writes a version of this included with Windows for the desktop and there are multiple 3rd party commercial and open source clients available.

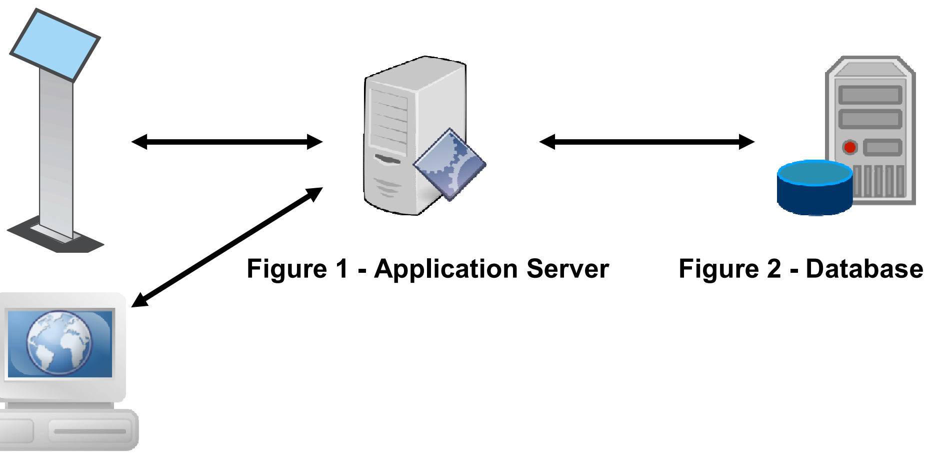


**Figure 1 - Application Server** **Figure 2 - Database Server**

## 2.2 Product Features

Product features are designated as either required or optional. Required features are a must for inclusion in this release as they are deemed essential to the Kiosk's core purpose. Optional features are desired features and will be implemented in this release only as time permits.

### 2.2.1 Required Features

1. Patient Welcome Screen
2. Patient Identification
    a. Manual (on-screen or physical keyboard)
    b. Machine-readable (magnetic stripe, barcode)

        c. Biometric (fingerprint)

3. Appointment Confirmation
    a. Confirm provider
    b. Confirm time
4. Demographic/Insurance Confirmation
    a. Confirm address & contact info
    b. Confirm insurance plan
        i. Display card images, if available
5. Electronically sign required forms not on file or expired
    a. On-screen signature capture
    b. Dedicated signature capture pad (Topaz)
6. Collect co-payment and any balance due
    a. Option to allow patients to override amount to pay
    b. Swipe credit card
    c. Real-time processing with chosen gateway
    d. Electronic receipt signing
        i. On-screen signature capture
        ii. Dedicated signature capture pad (Topaz)
7. Wayfinding
    a. Optionally display map to direct patient based on appointment
8. Multiple language support for screen prompts
9. User-configurable options
10. Web-based dashboard

### 2.2.2 Optional Features

1. Additional biometric patient identification
    a. Facial recognition
    b. Palm vein recognition
2. Registration of new patients
    a. Scan & OCR/BCR driver's license
    b. Scan & OCR insurance card
    c. Identity verification (challenge questions)
3. Update of demographic/insurance information
4. Verbal screen prompts
    a. Recorded voice
    b. Text-to-Speech
5. User-replaceable graphical elements

6. Walk-in patients
7. Insurance eligibility verification
8. Targeted on-screen advertisements

Enabling this functionality is data synchronization between the Kiosk software and the practice's software used for maintaining patient accounts and appointments (typically, practice management software, otherwise known as PMS). This synchronization can be accomplished in multiple ways and is documented in Section 3.A.

## 2.3 User Class & Characteristics

The Kiosk is designed to lessen the burden of front-desk staff by eliminating the need for them to spend time with patients whose check-in process is straightforward and requires no special handling by practice staff. For this version of the Kiosk, those patients are the ones without any changes to demographic or insurance information and who have a scheduled appointment.

Patients are the first class of user targeted by the Kiosk software. However, practice front desk staff and their managers are also a targeted user class by virtue of the fact that there will be a Web-based dashboard for front desk staff to monitor patient check-ins and management reports for supervisors to monitor Kiosk utilization and effectiveness.

It's imperative that the hardware by as unintimidating as possible so as not to deter patients from walking up to the Kiosk. While we are not manufacturing our own hardware, we can limit the number of peripherals that we will require our hardware partners to accommodate and therefore present a less cluttered appearance.

The patient-facing features of the Kiosk software must be uncomplicated so as not to create frustration in order that the patients will want to use the Kiosk again on subsequent visits. Buttons must be large and spaced apart enough so less agile patients will have no trouble choosing desired options. Text must be large enough to read by those visually challenged.

The Kiosk software must run full-screen, hiding the underlying Windows interface for both security and usability reasons, and exiting the software must be restricted to practice personnel only.

The Kiosk monitoring software must be readily accessible by any authorized user within the practice. This will include front desk staff and their supervisors. It must promptly update itself so users are notified of new check-ins in a timely manner and the information relayed from the Kiosk software must clearly communicate the actions taken by the patient and the result of the check-in process.

## 2.4 Operating Environment

The Kiosk application software will run on a range of hardware platforms utilizing a range of versions of Windows. As a 32-bit application targeting Microsoft's .NET platform, it will be usable under Windows XP or later and be able to run on Windows Embedded and Windows POS iterations of those operating systems for lower cost, a smaller footprint, and better integration with enterprise networks.

The server component for Web-based Kiosk monitoring will run under Microsoft's IIS Web server and utilize ASP.NET.

While the .NET platform will provide for neither the smallest nor the fastest possible executable programs, the development time for the initial release as well as all future version will be significantly less than if using a lower-level programming language such as C++. This trade-off is acceptable given the desired performance characteristics of the applications. In other words, speed of program execution will be more than acceptable.

The database server software will be Microsoft SQL Server, which will run on Windows Server 2003 or later. As of this writing SQL Server 2005 is the current production release, however the release of SQL Server 2008 is imminent. We will ensure compatibility with both versions, however if suitable an on-premises SQL Server is not already installed and available for our purposes, we will install a freely redistributable of SQL Server 2008 Express Edition.

While the SQL Server software and even IIS are capable of running on the Kiosk hardware itself, we strongly recommend they run server-side. Each can run on the same server or separate ones and we do not require a dedicated server for our purposes. Both IIS and SQL Server can run on either a physical server or a virtual one.

Storage requirements will vary greatly depending on the number of patient check-ins and whether or not card images are being scanned and saved.

## 2.5 Design and Implementation Constraints

The primary design constraint is the mobile platform. Since the application is designated for mobile handsets, limited screen size and resolution will be a major design consideration. Creating a user interface which is both effective and easily navigable will pose a difficult challenge. Other constraints such as limited memory and processing power are also worth considering. SplitPay is meant to be quick and responsive, even when dealing with large groups and transactions, so each feature must be designed and implemented with efficiency in mind.

## 2.6 User Documentation

With ease-of-use being the primary goal and with installation to be performed only be adequately trained staff, user documentation should largely be unnecessary. Documentation will be limited to documenting the user options to clearly state their effect, to documenting the usage of the Web-based dashboard, and to the data interface requirements.

## 2.7 Assumptions & Dependencies

### 2.7.1 Time Dependencies

As with most things, time is a limited commodity. All required features will be implemented regardless of time requirements. Optional features will be omitted if development times exceeds expectations or added based on priority if time permits after all required features have been added.

### 2.7.2 Hardware Dependencies

The minimum hardware requirements to operate the Kiosk software will be a Windows computer using an Intel Celeron processor with 1 GB of RAM and 120GB hard disk. It will also contain a magnetic stripe reader. However, this hardware is likely inadequate to run the Kiosk software locally for some of the features (e.g. form signing) that are more hardware-intensive. If the software were to be run on a server and accessed from the kiosk via the RDP protocol, however, this minimal hardware configuration will be more than adequate.

For a more robust user experience that fully makes use of all features and provides some assurance that future versions will also perform adequately, a computer with at least a Core 2 Duo processor and 3GB of RAM should be used.

If running from a server over RDP, please allow for at least 1GB of memory for each kiosk that will simultaneously run the application in the server's memory space.

Additionally, please allow enough server memory and processor power to accommodate SQL Server and IIS according to Microsoft's specifications and consider whether they are running on the same server or separate ones.

Various software features will require different hardware. The software must adapt to available hardware so that UCoA customer's do not have to purchase hardware they never intend to use.

### 2.7.3 External Dependencies

The Kiosk software requires access to the data maintained in the PMS or equivalent system in order to confirm patient identities and appointments. The exact requirements for how this data will be obtained is available in Section 4.4 (Communication Interfaces)

Payment Gateway API

While the payment gateway will be selected only after this project has begun, it is known that there will need to be an interface with at least one. There is no industry-standard API required, so each payment gateway will require a proprietary interface via their published API. For this reason, it is likely only 1 gateway will be supported with the initial release.

POS.NET API

Microsoft POS.NET interface will enable us to support various hardware devices (magnetic stripe readers, barcode scanners) using the same programming API regardless of manufacturer. This will simplify the support of wide variety hardware within each device type category and in some cases will provide built-in support of these hardware devices over the RDP protocol without required 3rd party USB redirection software.

ODBC/OLE DB/ADO.NET

These open database connectivity standards will permit us interface with most databases utilized by PMS in order to facilitate data transfer between them and our own SQL Server database.

TWAIN Scanning Interface

For scanner support we will utilize the TWAIN API in order to support the largest number of scanners as most desktop scanners include a TWAIN driver.

Innovative Card Scanning SDK

Used for Optical Character Recognition (OCR) of driver’s license and insurance cards and Barcode Recognition (BCR) on driver’s license with 2D (usually PDF 417) barcode.

USPS Web API

This will be used if the patient edits their address to determine that the change is accurate and complete so that bills and statements can be delivered to the address.

# 3. System Features

The Kiosk software specification has identified 2 categories of features; required and optional. The required features will be implemented in this release regardless of time constraints as they are deemed essential to the usefulness of the software.

Optional features will be implemented only if time permits and the decision will be made well into the development cycle as the implementation of required features nears completion. While some of the optional features will ultimately provide a significant enhancement to the application, they are not essential for the software to be useful.

## Core Features

While not defined as core features in and of themselves as they do not related to patient or staff interaction with the Kiosk application, there are certain programming requirements that are essential to each screen of the kiosk:

- They must have an inactivity timer that will cause the application to revert to a default “Home” screen after a certain amount of time has expired since the patient last interacted with the Kiosk. This will help limit the inadvertent exposure of personal information to another patient should the patient walk away mid-check-in.
- They must change all field and button labels according to the preferred language of the patient.
- They must utilize a customer-replaceable background graphic.
- They must occupy the full screen with no window decorations (border frame, title bar, minimize or maximize buttons).
- They must be “always on top” to prevent the Windows taskbar bar from appearing above them.
- They must provide a way to return to the home screen, abandoning the check-in.

Most of the core feature related to the Windows client software with which a patient interacts. However, there is also a server-based component that will enable practice staff to monitor Kiosk activity and utilization from anywhere they can access the internal Web server that will host the software.

The Web-based server component will run under Microsoft’s IIS Web server as it will be written using the ASP.NET framework. It will communicate with the database server hosting the Kiosk database to display and report on the Kiosk activity.

Conversely, the client software will run as a Windows forms application. After considering the possibility of writing this as a Web application a well so the software would require only a low-bandwidth connection for operation over WAN connections, that alternative was rejected due to limitations inherent in running software inside a Web browser.

While the software could run full screen, low-level access to hardware such as scanners, signature pad, and barcode readers would still require specialized applets (browser plugins or Java applets) in order to be accessible. Also, a Web-hosted user interface would not be as robust and screen updates would be slower.

At this time we can create a richer user interface with better response time and greater hardware support with less effort and therefore a shorter development cycle by using Windows forms. And for low-bandwidth support we can use the RDP protocol and 3rd party support for USB direction to support local hardware.

## 3.1 Patient Welcome Screen

This screen will be the “Home” screen and will appear upon startup of the application. It will also be the screen to which the software will return to after a check-in is completed, whether successful or not. This screen will contain a background graphic that can be customized for each practice in order to brand the Kiosk by including a practice logo and utilizing a color scheme consistent with their Web site, signage, business cards, etc.

This screen will then have a “Start” button if only a single language (English) is supported, or multiple buttons allowing patients to indicate their preferred language.

## 3.2 Patient Identification

In order for the Kiosk to identify the patient, it must gather identifying information first and then perform a search. The unique identifying information consists of the patient’s first and last name and their date of birth. This information can be obtained via on-screen prompts and manually data entry by the patient or it can be “read” from a driver’s license or, at least partially, from a credit card.

A driver’s license can be scanned and the front side’s information converted to text via optical character recognition (OCR). When available, the reverse side can be scanned for a barcode and it decoded to extract the name and date of birth.

Alternatively, some driver’s licenses contain a magnetic stripe that contains the information which can be read via the same magnetic stripe reader used for payments. And finally, a credit card’s magnetic stripe can be read and the cardholder’s name extracted. Note, however, that the name may not be that of the patient, may not be exactly the same as record in the PMS, and certainly will not contain a date of birth. In the case of a credit card being used for identification, a patient will be prompted for his or her date of birth before a search will be performed.

### 3.2.1 Manual Entry

1. Patient enters last name then first name using on-screen keyboard
2. Patient enters date of birth using date selection screen

### 3.2.2 Magnetic Stripe Swipe of Driver’s License or Credit Card

1. Driver’s license or credit card is swiped by patient using kiosk-attached magnetic stripe reader
2. Magnetic stripe data is parsed for first & last name and, when available, date of birth
3. When date of birth not in magnetic stripe data, date selection screen is presented to patient

### 3.2.3 OCR/BCR of License

1. Patient scans driver's license using kiosk-mounted scanner
2. Card image is sent to OCR/BCR engine to extract
3. Extracted data is parse of patient first & last name and date of birth

#### 3.2.4 Biometric (fingerprint)

Note: This option will only work if fingerprint is either pre-registered using a desktop application (after patient is positively identified) or if patient self-identifies at kiosk first and then goes through a fingerprint templating procedure.

1. Patient places finger on hardware-attached finger scanner
2. Finger profile is compared to limited set of previously-templated fingerprints due to the time required for comparison. This limited set will likely be of scheduled patients and possibly further limited to a small time range around current time.

### 3.3 Appointment Confirmation

This step will help ensure that the patient indeed has been scheduled for an appointment on the date and at the time as well as for the provider expected. It can also be used to verify that a patient is checking in where expected.

For example, in a multi-location practice the patient may believe that the provider is at a location at which the patient frequently sees him or her or the provider is a different one than the one the patient typically sees. By checking the scheduled appointment against the kiosk's known physical location we can verify the correct check-in location and inform both the patient and staff when it is wrong.

Furthermore, by knowing the appointment we can determine the correct co-payment amount (since it is typically different for primary care providers than for specialists) and route the patient to the next location within the practice. This latter step is useful for large facilities with a central check-in location but individual waiting areas closer to the provider's examining rooms.

### 3.4 Demographic/Insurance Confirmation

A key task of any check-in process, manual or automated, is to confirm that the essential patient information on file is current and accurate. Here the Kiosk can actually outperform a human by never failing to confirm it and by displaying it on-screen for visual confirmation allowing the patient to see what is currently on file.

The essential elements are those required for accurate billing and collection of insurance and patient due portions. This includes the patient's address, phone numbers, e-mail address, and insurance plan information. It may also include emergency contact information that, while not for billing and collection, is essential to have on file.

A user option will allow the enabling of editing these elements as well. Data changes will be applied within the Kiosk database and can be replicated to the PMS by either a "copy and paste" manual procedure or via an API (e.g. HL7) for automated data replication.

If insurance card images are available in the PMS, displaying them will significantly improve the patient's recognition of current vs. outdated data. However, it is not perfect.

For example, a Medicare card may be unchanged but the patient has opted into an Advantage plan whereas previously had been covered by the traditional fee-for-service option. Therefore, this option is not a replacement for validating insurance benefits via real-time validation. Real-time verification of presumably unchanged coverage is found on our Optional Features list.

## 3.5 Electronic Form Signing

All patients must have on file an acknowledgement that they have been provided a copy of the practice's HIPAA Privacy Policy. While the form itself is not universal, the requirement is. Nearly as universal is a notice of the practice's financial policies. And to a lesser extent, practices may require a Consent for Treatment.

Here, too, a Kiosk can be more efficient than a human. A kiosk can reliably check if a form is on file and if it was signed within a reasonable time period. For example, even if a signed HIPAA notice is on file, the practice may want it signed again if it's been more than a year. The Kiosk won't forget to prompt the patient and can either alert staff if it was skipped or altogether prevent the patient from continuing with the check-in.

The Kiosk software will be capable of displaying a copy of the practice's existing forms. This will be done by taking an electronic copy of the form in the TIF image format and loading it into the database by way of or current POWERsign software. Using POWERsign, we can also define additional data elements from the patient record or system data that are to appear on the form and their location as well as the location of the signature.

At signature capture time, the blank form will be loaded, pre-filled with the dynamic data elements, and displayed on the computer monitor. If screens smaller than 15" are to be supported, zoom in/out and scroll buttons should enable the patient to enlarge and pan around the form.

Utilizing either a stylus and the computer's touchscreen or an externally mounted signature capture pad, the Kiosk software will capture the patient's signature. Navigation forward should either be blocked pending the signature or a warning should appear if the patient attempts to advance in the workflow without signing. This could be a system option so each practice can decide for itself.

Finally, the patient will be offered the option to print the form after it's been signed. The completed form will then be printed to the system's default printer, which may be mounted inside a kiosk cabinet, located on a table or shelf nearby, or at a nearby desk of an practice employee (e.g. behind the registration desk).

If the printer is not nearby the patient will need to be informed where to pick up their copy and if it is nearby, the patient should be reminded to take their copy to protect their privacy.

## 3.6 Payment Collection

Once again, the Kiosk trumps humans. The Kiosk is not embarrassed or hesitant to ask for money owed and is immune to sob stories. However, it can not enforce collection of monies owed; it can only abort the check-in and refer the patient to a human.

The Kiosk will be capable of retrieving any prior patient due balance from the PMS as well as determine whether a primary care or specialist co-pay applies (based on the specialty of the provider associated with the confirmed appointment). It will prompt the patient for the total amount due as the sum of both. A system option will dictate whether the patient can alter the payment amount. Options could include a simple yes/no and/or may define that the patient must pay a minimum amount determined by what is owed. We may need to allow for unique, custom formulas for each practice where some may prefer that at least a portion of a prior balance be paid as well as current charges or just that the patient not fall further into debt.

Once a payment amount is finalized, the patient will use a hardware-attached magnetic stripe reader to swipe their credit card. Card data will be sent in real-time to our selected payment provider (to be determined early in the development stage) and the results parsed. This will allow the Kiosk to prompt for another payment method should a credit card be declined or prompt for a signature (electronically captured) after displaying a receipt.

Printing a copy of a receipt will be a patient option following signing. The receipt will then be printed to the system's default printer, which may be mounted inside a kiosk cabinet, located on a table or shelf nearby, or at a nearby desk of an practice employee (e.g. behind the registration desk).

If the printer is not nearby the patient will need to be informed where to pick up their copy and if it is nearby, the patient should be reminded to take their copy to protect their privacy.

## 3.7 Wayfinding (facility maps)

While not intended to be a standalone wayfinding system, which would allow a patient to search for a destination and be presented with a map and/or directions, the Kiosk's wayfinding system will be directly tied to the patient's appointment.

The last step of a check-in when wayfinding is present is to display an on-screen map to inform the patient to where they should proceed. Perhaps the Kiosk is located inside the main doors of a large, multi-story practice. Such a practice likely has waiting areas throughout the building for each department (e.g. radiology, cardiology, family medicine). Based on the physician and/or the physician's specialty associated with the confirmed appointment, the appropriate map can be selected.

A print option would allow the patient to print the map. The printed map may also include step-by-step walking directions that could include instructions to take an elevator to another floor as well as turn-by-turn hallway directions.

The directions will be printed to the system's default printer, which may be mounted inside a kiosk cabinet, located on a table or shelf nearby, or at a nearby desk of an practice employee (e.g. behind the registration desk).

If the printer is not nearby the patient will need to be informed where to pick up their copy and if it is nearby, the patient should be reminded to take their printout.

## 3.8 Multiple Language Support

Recognizing that our current client base serves a diverse patient population, supported multiple languages is essential to assuring the Kiosk can be used by the largest number of patients in order to maximize utilization. A necessary restriction of features will affect utilization as will some patients' willingness to use the technology. These are unavoidable.

However, there is no need to artificially restrict access by not being able to use additional technology to expand its reach to patients who otherwise may not feel comfortable using a Kiosk in their non-native tongue but are perfectly will to accept it otherwise.

With the availability of online language translation tools, the time required to create an additional set of language prompts can be minimized. Once an initial translation is performed, native speakers of the language can provide a final edit to ensure that the meaning of each phrase is clear and in common use.

The language translation themselves should be stored in a way that can allow editing by the end user. This will allow each practice to alter our prompts as they see fit, whether it's due to a poor translation or simply that they believe it is unclear to their patients.

The phrases can be stored in a database table or XML file. As they will only be loaded once at the start of each check-in, the performance of either will be acceptable. No special indexing is required and the entire translation will be loaded regardless of how it is stored.

Storing in an XML file will allow editing using any text editor and may permit bulk translation. Storing each language in its own XML will easily allow us to determine available languages without taking a database hit by either putting the language files in their own folder or defining our own file extension. It will keep each language autonomous and allow the easy transfer of language files from one Kiosk to another, especially across practices or from UCoA to a practice without having to create an export/import utility.

A portion of the XML file may look like this (formatted for readability):

```
<Language ID=”Español”>
      <Snippet ID=”LabelLastName”>
            Apellido
      </Snippet>
      <Snippet ID=”LabelFirstName”>
            Nombre
      </Snippet>
      <Snippet ID=”LabelSwipeCreditCard”>
            Pase su Tarjeta de Crédito
      </Snippet>
</Language>
```

The XML contains not only the language translations for each and every system prompt, but the label for the button on the welcome screen as well. Everything is completely self-contained and can easily be updated and transferred between installations.

## 3.9 User-Configurable Options

Because no two practices are alike, it's imperative that the system be able to adapt the requirements of each as much as possible. These options can relate to the look and feel as well as functionality and workflow.

Options should include the ability to disable certain pieces of the workflow entirely (e.g. wayfinding or form signing), alter the flow itself (e.g. sign forms before or after taking payment), or alter the rules of background processes to be performed (e.g. address verification) or when the workflow may advance (e.g. require a form to be signed).

Other options may be used to determine which hardware features are available, though the preference would be to check for the presence rather than relying on a user option; or at least verifying it if the user tells us to use it. However, sometimes a device may be available but we don't want to use it.

For example, while a magnetic stripe reader will no doubt be a standard feature of all hardware configurations, we may not want to prompt the patient to swipe a credit card or driver's license for identification (perhaps because the license for most patients doesn't contain a magnetic stripe and the practice doesn't want to have patients deal with the limitation of swiping a credit card for identification).

Or perhaps a fingerprint reader is standard hardware because it's inexpensive enough to add at manufacture whereas performing a field upgrade may not viable. Should the customer some day want to add this feature they would not need to replace the hardware and yet the feature could be deactivated until they are ready to implement it.

Additionally, we may install all available language files so a client can choose which ones are pertinent to them. A system option could allow the easy activation/deactivation of these language files

With enough options, two Kiosks may look and act vastly difference to the point that it could be difficult to recognize that they are running the same software.

The options will appear upon application startup just prior to the welcome screen. However, this should only be accessible by passing a special command line option, or at the very least allow a command line option to bypass the options screen. This is necessary so that a Kiosk can be configured to automatically go to the welcome screen upon startup. This is most important where the Kiosk may startup unattended, such as after a power outage.

## 3.10 Web-based Dashboard

Providing for a way to access the activity and utilization of the Kiosk is an integral part of the system. Front desk staff need to know who checked in and the result of the check-in. Management needs to know how the Kiosks are being used and measure their effectiveness.

We've chosen to implement the dashboard as a Web-based application. While this imposes a requirement of a server running Microsoft IIS (with ASP.NET), it allows the application to be accessed by anyone who can reach the server by IP address or machine name and has the appropriate permissions. Unlike the Kiosk application itself, there is no need for hardware integration or instantaneous screen updates, so accessibility and usability are the only requirements and make a Web application the ideal platform.

The default page for the dashboard will display the most recent Kiosk activity. This page will auto-refresh so no user action will be required to display the newest check-ins. Check-ins will be color-coded to indicate their result; white for in-progress check-ins, red for check-ins requiring staff attention (e.g. patient refused to pay), green for check-ins that completed as expected, and yellow for check-ins that *may* required a staff member's attention (e.g. patient altered payment amount to pay less than amount requested).

Each check-in will display the time it began, the time it ended, the patient's name, the appointment for which the patient checked in (time & provider), and all other pertinent activity and what, if any, errors occurred. Pertinent activity may include address or contact information changes, forms signed, and amount paid. It could also include the prior balance and co-payment amount for which the patient was prompted, the wayfinding map displayed, and whether the patient requested a copy of signed forms, receipts, or maps.

Errors that could occur and cause an check-in to be aborted (and displayed on the dashboard in red) may be that the patient moved (assuming address changes are not allowed by user option), has new insurance, didn't sign required forms (if it's a requirement by may of user option), or didn't pay.

Warnings that could occur (shown in yellow on the dashboard) may include a patient not signing a form (assuming they are allowed by user option to continue checking in) or not paying in full for the amount requested (assuming the patient is allowed by user option to change the payment amount).

The system will need sub-pages, accessible from the dashboard, that link to details of the check-in. These sub-pages will allow the staff to view a payment receipt and issue a refund (e.g. if the patient leaves without being seen), view signed forms, and see the patient's check-in history.

Kiosk utilization reporting will help managers improve Kiosk usage and understand where improvements need to be made. A summary report such as check-ins by result for the week, month, and year will give an idea of whether Kiosk usage is increasing or decreasing, whether payment collections by the Kiosk are as expected, how often the Kiosk is being used to gather minor demographic updates, and whether patients are able to complete the check-in process at the Kiosk.

Additional reporting can be used to discover the effectiveness of having patients guided to a Kiosk. If a staff member is made available to help patients for a period of time or for certain times of the day, reporting the number of check-ins and how many were successfully completed by day of the week and time of day can be cross-referenced to that employee's schedule to determine the efficacy. By comparing the financial benefit gained to the cost required to obtain it, the practice can determine whether it's worthwhile to continue expand or reduce that service. The practice may find it beneficial enough to add a new staff member, perhaps at a lower hourly wage to provide an even greater positive result.

Comparative reports will enable managers to determine if Kiosks at different locations are achieving similar results. Of course, as much information as possible should be provided to help determine why this may be. Raw numbers alone may not be enough. Just because fewer check-in are performed doesn't mean a Kiosk is underused. It could simply be the result of fewer appointments at that location. For a raw number like that to be meaningful we would need to know the number of scheduled appointments. This could be acquired from the PMS.

Comparing the percentage of successful-to-attempted check-ins is more useful with a large enough data sample. However, in order to act upon that information it's important to understand the patient demographics. Perhaps the providers being served by the Kiosk have a larger number of elderly patients or a greater number of indigent ones. The former could be determined from the information we gather during check-in and the average age of those checking in can be reported, but the latter would be harder to determine by the software and would require the user of the data to make their own conclusion.

## Optional Features

We have identified a number of features that we have deemed useful to the Kiosk but not critical to its initial success. These features will have varying levels of positive contribution to the acceptance of the Kiosk by practices not among the early adopters and will be prioritized based on the time available vs. the time required for implementation as well as their intrinsic value to the application.

In other words, one or more less useful features may be added prior to a more useful one if the latter cannot be implemented in the remaining time until deadline whereas the less useful one(s) in all or part can.

In no particular order, here are the optional features that we have identified.

## 3.11 Additional Biometric Patient Identification Methods

### 3.11.1 Facial Recognition

Using a built-in Webcam, we will be able to automatically locate a patient's face in the field of view, capture an image, and create a digital profile of it. Using that profile we can search a database of previously-registered profiles to find a match to an existing patient.

Profiles can be pre-registered using either a desktop application by practice staff following positive patient identification (e.g. by photo ID) or one can be created at the Kiosk after a patient has identified him- or herself using one of the other available methods.

### 3.11.2 Palm Vein Recognition

This technology is similar to fingerprint recognition. The primary advantage of this over that is the lack of a requirement to make physical contact with the scanning device. This means one less surface that can be the source of germ transfer and therefore would be have to be treated with any antimicrobial chemical.

## 3.12 Registration of New Patients

This feature would enable patients at the practice for their first visit to complete some or all of the registration process at the Kiosk. Because this process can be long and arduous, we would want to make use of all reasonably available technology to automate as much of the process as possible. Yet it should also be simple enough so that the requirement for staff intervention will be limited.

For the registration of new patients to be useful to the practice, the acquired data should be sent to the PMS (and perhaps the EMR) automatically. The most universally-accepted method for this is HL7 messaging. Without an automated data transfer method the practice would be forced to copy and paste a large number of data items.

While it's conceivable that we could reverse-engineer the database updates required to manually add the required information to the PMS and EMR databases, the labor costs and possible errors would not make this a worthwhile venture. Furthermore, there is little to no possibility that the PMS or EMR vendor would approve of this and would put the practice in a terrible predicament with those vendors.

Practices making use of this feature would need to consider the number of new patients that would use the Kiosk to accomplish registration and the time required. While it would free practice staff to some extent, there would still be staff involvement both at the Kiosk and at the registration desk. Furthermore, as the Kiosk will be in use for far longer than for a simple check-in, depending on utilization additional Kiosks may be required, adding yet more to the cost of providing this feature.

### 3.12.1 Scan & OCR/BCR Driver's License

This technology will enable to us to use an attached duplex scanner to obtain an image of both the front and back of a driver's license in a single pass. We will then use an SDK from Innovative Card Scanning (ICS) to automatically locate and decode an available 2D barcode. When not available (damaged or not present), the SDK can be used to OCR the front of the license. The data returned by the SDK will include the name, date of birth, and address. This data will be used to begin the creation of the patient record. However, if no barcode is found, address verification should be performed to help ensure that the OCR results were accurate, even if the patient is asked to confirm them.

### 3.12.2 Scan & OCR Insurance Card

Similar to the scanning and OCR/BCR of the driver's license, this function will also make use of the duplex scanner and ICS SDK. Similarly, the data will be extracted and presented to the patient for confirmation/edit.

At this point there is some debate on how to ensure that the recognized data, despite being confirmed by the patient, is both accurate and current. Ideally we would perform a real-time insurance eligibility check. Practically speaking, this would require a way to determine which of our electronic payers to which to send the request.

With over 300 possible payers the list is too long to have a patient scroll through it and select the most appropriate. Perhaps the SDK can identify the template used and that in turn can be used for a one-to-one association with one of our payers. However, if there is a one-to-many correlation or if an automated updated of the SDK templates results in a change for which we have not accounted as yet, no association can be made.

Initially, it may be best to simply capture the information and have the practice perform the eligibility verification using another method (payer Web site, clearinghouse, or within the PMS).

#### 3.12.3 Identity Verification

If we are to allow patient self-registration at the Kiosk it would stand to reason that we should provide a method of ensuring that the patient is who he or she claims to be. When registering with a staff member a photo ID is typically presented and a comparison made between the picture on the ID on the person standing before them.

Similarly, we could scan a driver's license and extract the photo, then compare that photo with an image of the person standing in front of the Kiosk by capturing it via a built-in Webcam.

Alternatively, private databases can be used to create challenge/response questions/answers (knowledge-based authentication) to confirm an identity. The information required to answer the questions we would present to a new patient could not be answered even if that person were an identity thief in possession of the real person's wallet.

Various companies have built these private databases and make them accessible via an API. Many patients will be familiar with the type of questions as they are similar to what they would have seen if they accessed their credit history or FICO score on a Web site.

### 3.13 Update of Demographic/Insurance Information

This task would be largely similar to a new patient registration and has the same drawbacks as well as benefits. The biggest differences are that only the changed data needs to be collected and we do not necessarily need to scan a driver's license.

While a driver's license is scanned for a new patient registration in order perform OCR/BCR to acquire the data, a side benefit is that the license is stored and eliminates the requirement that the staff photocopy or scan it themselves.

This is less important when there is just a change to demographic data. Yes, it may be helpful to have the new image with the updated data. There is also a chance the license hasn't been updated yet. Therefore, the update of an address will probably best be accomplished by allowing the patient to entering it using the on-screen keyboard and bypass the requirement to scan the license again.

However, a change of insurance almost certainly has resulted in the issuance of a new insurance card and it's imperative that the practice have a current copy of the front and back. Therefore, for new/updated insurance plans we will perform the same function as if it were a new patient registration.

### 3.14 Verbal Screen Prompts

Recognizing that there is limited screen real estate with which to provide patient instructions, verbal screen prompts will enable us to provide more detailed instructions simultaneous with the start of a new task in the workflow.

These screen prompts can be implemented one of two ways and can be used in combination, where some prompts are delivered via one method and others via another.

#### 3.14.1 Recorded Speech

Our end users would record sound files using a microphone. These sound files will be specifically named so that when a particular application task is started we can check for its existence and play it, if present.

#### 3.14.2 Text-to-Speech

Where a sound file does not exist, a text passage can be converted to a sound file dynamically and played. For best performance, we would check for these text passages and application startup and convert them to sound files in advance. This will allow the application to simply locate the sound file and play it without any additional processing overhead.

Microsoft Windows has a text-to-speech SDK, however the engine and/or voices leave something to be desired in terms of quality. They simply sound too robotic.

Implementation of this feature should be delayed until a better speech engine can be found at reasonable cost.

### 3.15 User-Replaceable Graphical Elements

The initial release includes the core feature of replaceable background graphics for both the welcome screen and all of the others. This optional feature would extend customization to include button backgrounds/shapes, animations (e.g. card swipe, signature capture), and other graphical elements so practices can custom it as they see fit or we can advance the aesthetics independent of the underlying code. It would also enable us to adapt the product to match the underlying hardware. For example, the magnetic stripe reader could look different from one hardware platform to the next and an animation depicting the swiping of a card would be improved if it were to match.

### 3.16 Walk-In Patients

The initial release of the Kiosk will not support walk-in patients because there are special requirements that must be handled. It is not as simple skipping the appointment confirmation. We would not know the correct co-payment nor the proper wayfinding map.

Handling walk-in patients requires prompting for the visit reason. This step may include providing a list of possible reasons with each reason containing data necessary for the remainder of the workflow. Some reasons (e.g. head injury, chest pain) may also result in the immediate abort of the check-in and generate an alert to a staff member.

Certain reasons may not specifically associate with a provider (e.g. allergy shots or blood draws) and may have their own rules regarding co-payments or collecting payment from private pay patients. For others we may need to identify the provider for proper co-payment collection.

## 3.17 Insurance Eligibility Verification

This was partially covered in Section 3.12.2. There we mentioned the difficulty of performing this on new patients and new insurance plans. However, there is one instance where this is a straightforward process; verifying that an existing plan is still active after the patient claims it is.

In this case, the data on file will have enough information so we can associate a payer from the PMS with of our own using a mapping table. In only a rare number of cases will we not have an association once the table is defined. When we don't, it will typically be a rarely-seen payer. Even when it's not, the table mapping can be easily updated after the first instance by the practice.

We would provide a user option to enable this feature for just primary insurance or both primary and secondary. We would also allow the practice to specify how often to run a verification. This would enable the Kiosk to automatically verify eligibility only if the patient hasn't been seen (and insurance verified) within *X* number of days/months. The practice can therefore choose their desired balance between cost and relevance of the most recent report.

## 3.18 Targeted On-Screen Advertisements

There are 2 reasons this might be implemented; 1) providing a method for 3rd party advertises to provide meaningful ads for which they will pay by way of subsidizing the Kiosk; and 2) providing a method for the practice to provide their own ads to promote their own services in order to increase revenue per patient.

Ads will use unidentified patient data (age, gender) to target the right ads for the one-person captive audience. Subsequent releases of the product may tap into EMR data to further tailor ads based on medications being taken regularly, a chronic illness, or the reason for the current visit.

Ads can be full-screen videos that the patient must watch before advancing in the workflow or static graphics, animated graphics, or videos that appear on one of the workflow screens. This may be a practice option or would depend on whether a 3rd party ad provider is subsidizing the Kiosk and sets a requirement.

# 4. External Interface Requirements

## 4.1 User Interface

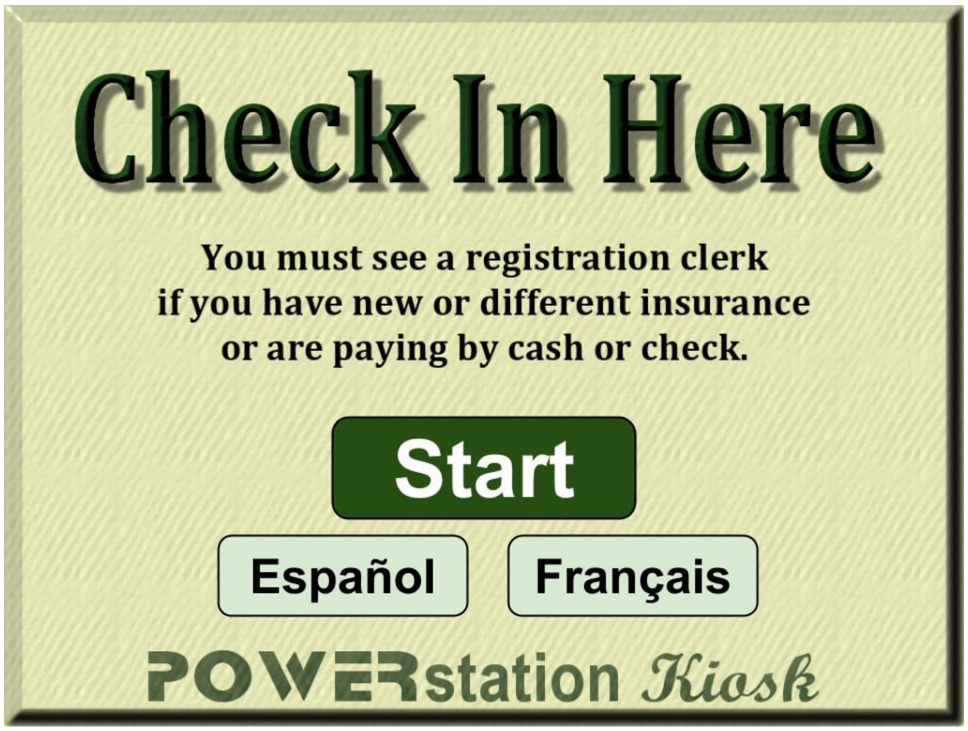
### 4.1.1 Welcome Screen

This is the screen that starts the patient flow. After the application launches it will display this screen and after the completion (successful or otherwise) of a patient check-in it will also revert to this screen.

This screen will also be displayed in response to the patient selecting a “return to home” button or after an inactivity timeout period should a patient stop interacting with the software for a period of time.

This screen will need to dynamically determine if multiple language translations are available and, if so, offer the patient a choice of languages (which will dynamically alter all screen prompts going forward).

This screen should be a single background graphic that is dynamically loaded upon startup, allowing it to be customized for each customer and possibly each kiosk. Only the language button(s) will be user interface elements.


Check In Here

You must see a registration clerk
if you have new or different insurance
or are paying by cash or check.

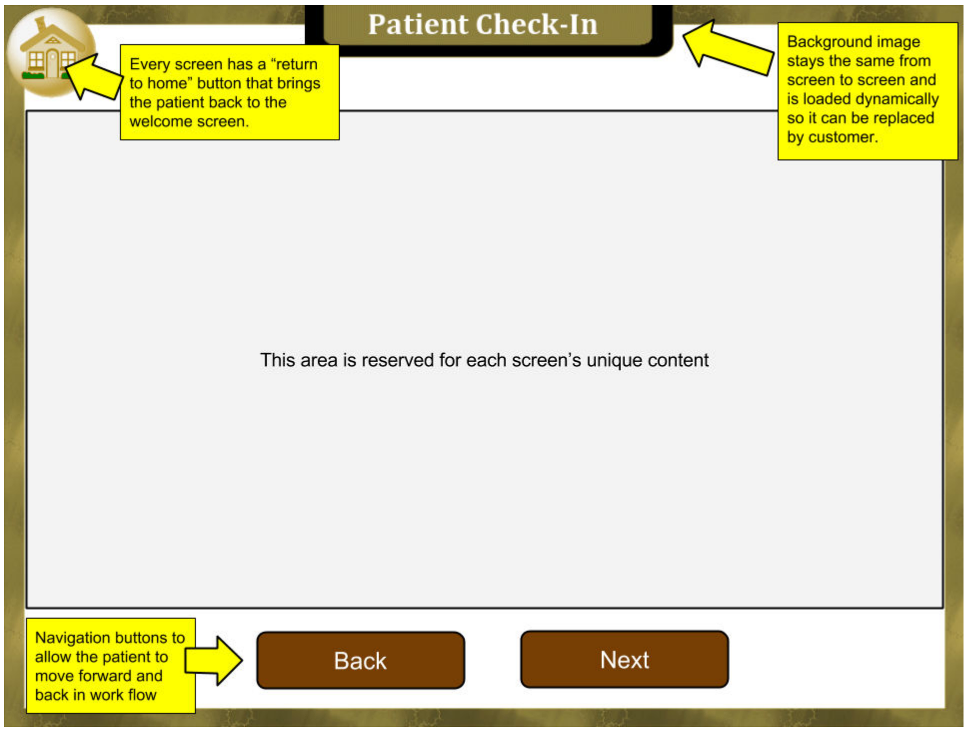
### 4.1.2 Standard Screen Layout

All screens will have the same general layout for consistency. Nothing would create an unusable system as much as one that requires the user to hunt for pertinent information each time the screen changes.

Large, easy to read navigation buttons will enable patients with visual acuity problems and/or less dexterity find and act upon them.

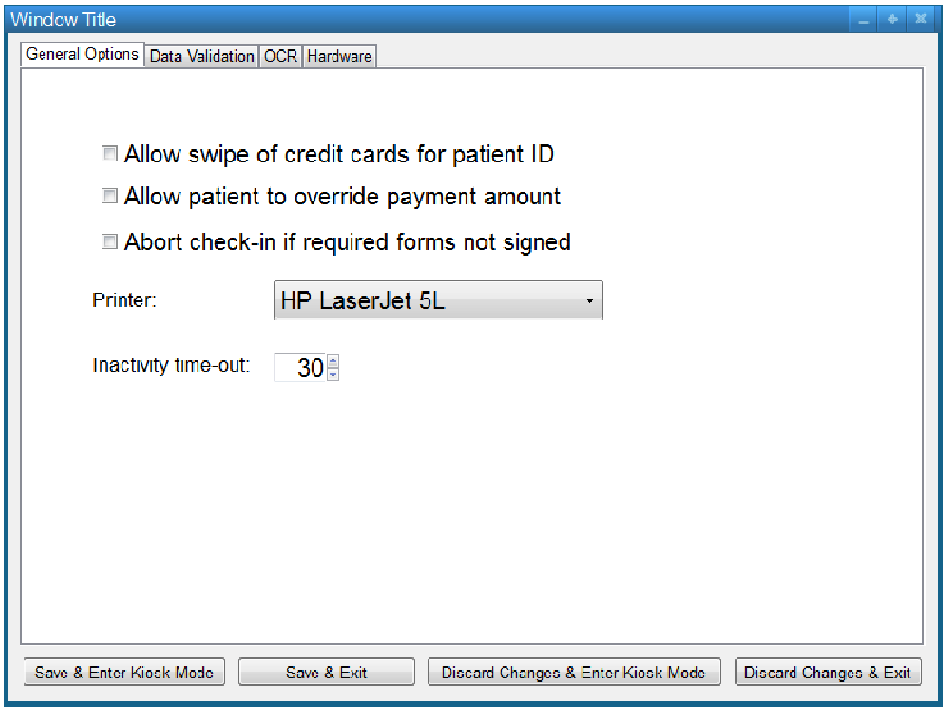
Each successive screen will largely change only in the center area, however it's possible that the navigation buttons may need to be hidden or made visible dynamically. For example, if a patient is editing their address, the center portion of the screen may show a keyboard and textbox (displaying the entered information) along with Accept and Discard buttons.

We may not want the patient to move forward or next in the workflow until after the data entry is either completed or abandoned as it is unclear what action should be taken if data entry is performed. What if a partial address is entered and then the Next button is used? Should it be updated? And should it then be abandoned if Back is used? Should we instead prompt the patient if data is changed regardless of action? These are all valid options and it may be necessary to test each of these via a usability study to determine if one is preferred over the others.


Patient Check-In
Every screen has a "return to home" button that brings the patient back to the welcome screen.
Background image stays the same from screen to screen and is loaded dynamically so it can be replaced by customer.
This area is reserved for each screen's unique content
Navigation buttons to allow the patient to move forward and back in work flow
Back
Next

### 4.1.3 System Options Screen

All



☐ Allow swipe of credit cards for patient ID
☐ Allow patient to override payment amount
☐ Abort check-in if required forms not signed

Printer: HP LaserJet 5L

Inactivity time-out: 30

### 4.1.4 Dashboard Screens

All

## 4.2 Hardware Interfaces

Hardware interfaces explained

## 4.3 Software Interfaces

Software interfaces explained

## 4.5 Communications Interfaces

Communications interfaces explained

# 5. Other Non-Functional Requirements

5.1 PERFORMANCE REQUIREMENTS
Performance should not be an issue because all of our server queries involve small pieces of data. Changing screens will require very little computation and thus will occur very quickly. Server updates should only take a few seconds as long as the phone can maintain a steady signal. The cost-division algorithms used by in application will be highly efficient, taking only a fraction of a second to compute.
5.2 SAFETY REQUIREMENTS
SplitPay will not affect data stored outside of its servers nor will it affect any other applications installed on the user's phone. It cannot cause any damage to the phone or its internal components. The only potential safety concern associated with this application applies to virtually all handset apps: SplitPay should not be used while operating a vehicle or in any other situation where the user's attention must be focused elsewhere.
5.3 SECURITY REQUIREMENTS
This application assumes that only the user or whoever he/she allows will have access to his/her Android handset. With that being said, only a Google email address is required to verify the identity of the user upon opening the app. Since it is not password protected, there is no method to authenticate the user's identity. This could only pose a thread if a user has set up PayPal functionality, however any transaction involving real currency must be authorized and confirmed before becoming final. The PayPal API provides all of the security checks needed to ensure that no fraudulent transactions occur.
5.4 SOFTWARE QUALITY ATTRIBUTES
The graphical user interface of SplitPay is to be designed with usability as the first priority. The app will be presented and organized in a manner that is both visually appealing and easy for the user to navigate. There will be feedbacks and visual cues such as notifications to inform users of updates and pop-ups to provide users with instructions.
39

SPLITPAY 40
Software Requirements Specification
To ensure reliability and correctness, there will be zero tolerance for errors in the algorithm that computes and splits expenses between group members. To maintain flexibility and adaptability, the app will take into account situations in which a user loses internet connection or for whatever reason cannot establish a connection with the server. These users will still be able to use the application, but any Bills, transactions, etc. posted while disconnected will be cached until the connection is restored.
Furthermore, the group leader also has the option to add members who do not own an Android phone and add transactions on their behalf. With SplitPay being ported solely for the Android platform, this software application has the advantage of being portable and convenient to use whenever and wherever. Overall, the app balances both the ease of use and the ease of learning. The layout and UI of the app will be simple enough that users will take no time to learn its features and navigate through it with little difficulty.
40

SPLITPAY 41
Software Requirements Specification
6. KEY MILESTONES

41

| Milestone | Deadline | Comments |
|---|---|---|
| SRS document | 2/8 | Abstract done by ian 31 so a presentation can be created |
| Finalized Interface Design | 2/19 | Designs should be done a week in advance of abstract for revisions |
| Finalized Algorithm Design | 2/19 | Designs should be done a week in advance of abstract because of bugs |
| SDD document | 3/3 | Abstract done by Feb 26 so a presentation can be created |
| ServerSetup | 3/10 | This should be done in the beginning of the implementation . phase to sync with other features |
| Interface & Algorithm Implementation | 4/13 | This should be completed two weeks in advance of complete project so app can be thoroughly tested and bugs fixed |
| Software Festival | 4/27 | Completion of project |

SPLITPAY 42
Software Requirements Specification
7. KEY RESOURCE REQUIREMENTS I

Implement the Familiarity with All team members Android Developers Potential schedule
interface Android GUI have Java community; published conflicts
implementation programming examples
experience Android emulators are
publicly available for
testing
42

| Major Project Activities | Skill/Expertise Required | Internal Resource | External Resource | Issues/Constraints |
|---|---|---|---|---|
| Create a Server application | Database systems & servers experience | Nick and Eric have general knowledge in this field | Internet | No physical server; limited number of team members with SQL/PHP experience |
| Sync the Server to the Application | Application networking with online servers | Nick and Eric have general knowledge in this field | Internet | Possible compatibility issues |
| Design the Interface | Design & usability experience; familiarity with Android GUI design | Josh has extensive digital design experience; Rick has experience with human-computer interaction (HCI) | Android Developers community; published examples | Schedule conflicts; physical constraints of handset screen resolution |
| Design an algorithm | Knowledge & experience designing | All members have taken data structures and | Android Developers community; | Potential schedule conflicts |

| | | | | |
|---|---|---|---|---|
| | mathematical algorithms | algorithms | published examples | |
| Implement an algorithm | Familiarity with the Android development environment | All team members have Java programming experience | Android Developers community; published examples Android emulators are publicly available for testing | Potential schedule conflicts |

SPLITPAY 43
Software Requirements Specification
8. OTHER REqUI”
A database for SplitPay calls for a server side implementation that holds information for the users, groups, bills, transactions, as well as all the relationships involved. The database will be using MySQL. The following provides an example of information that may be stored in the database:
• Users: ID, Name, phone number, email, PayPal username
• Groups: ID, Name, Members
• Bills: ID, Name, Owner, Group, Cost, Receipt, Date, Time
• Transactions: ID, Members Involved, Group, Date, Time, Amount
The server will be configured on a Linux platform, and through use of PHP will allow interaction and processing in conjunction with the database. Processes to be done on the server include: pushing/pulling data, updating data, and generating notifications.
43

SPLITPAY 44
Software Requirements Specification
9.APPENDIXA GLOSSARY
OFFLINE USER
A user that has been added to the SplitPay server but does not have a copy of the application. These users are contacted exclusively through email and/or SMS. An offline user may become a SplitPay user by downloading the application and claiming their account.
GROUPS
Groups in the context of the SplitPay application are a collection of users that can create Bills and Transactions amongst each other.
Bills
Bills are a method of distributing an expense among some or all members of a group.
TRANSACTIONS
Transactions are used for re-allocating debt to different members of the group. Transaction amounts may be positive or negative and may involve two or more members of a group.
DISBAND AND DELETE GROUP
Terminates the Group. All debts are left hanging and no final debt resolution notice is issued. The group is removed from the Groups screen.
STOP GROUP
Bills and transactions that add debt can no longer be posted to the group. The group enters the debt resolution state where transactions may only be applied to resolve all debts. The group still persists in the Groups screen.
STOP AND DELETE GROUP
The group is removed from the groups screen, and a final debt resolution notice is issued to all

members via email.
44

SPLITPAY 45
Software Requirements Specification
10. APPENDIXB PROJECT PR0P0sAL1
Group 1492: Software Application Proposal
Working title: SplitPay
Platform: Android (2.2+), predominantly Java programming
MOTIVATION
We are designing an application that facilitates the process of paying off shared expenses. In other words, we want to make it easier to pay for things as a group. This is relevant in many different situations, ranging from every day transactions (friends splitting the cost of dinner), to recurring payments (roommates paying rent), to a more professional setting (a team of professionals making business transactions). Depending on the number of people and/or the amount of money involved, these situations can get very complicated.
PROBLEM STATEMENT
The problems we aim to alleviate with this app include the following:
• Difficulty of calculating divided costs
• Unfair or imbalanced payments amongst individuals
• Lack of accountability for money owed
• Difficulty remembering payments already made, outstanding debts, etc.
The app we design will solve these problems and several more. Consequently, we will include the following functionalities:
• Automatic calculation of money owed by each member of the group
• Customizable formula for splitting costs
• Transaction history (including all payments made & costs incurred by each individual)
We will also add the following features, which extend beyond the simple task of dividing costs:
• Support for multiple transactions which can take place over extended periods of time
• Automatic notifications, alerting users of outstanding debts and/or other relevant info
• Transaction synchronization, keeping each group member up to date at all times
• PayPal integration, allowing for easier transactions
• Receipt storage, which allows the user(s) to save photos of receipts associated with particular purchases/expenses
• E-mail and print functions, allowing non-Android users to obtain copies of transactions
NOTE: Some of these features are not part of our core design and will only be added if time permits.
45

SPLITPAY 46
Software Requirements Specification
OBJECTIVES
Beyond implementing the above features, there are a number of additional objectives we must accomplish as a team. First and foremost, as a group we must familiarize ourselves with the Android platform, as this will lay the groundwork for future development. Additional objectives include the following:
Develop a timeline detailing each stage of development
Establish well-defined roles for each member of the group
Create detailed software specifications
Ensure each member is up to speed and completing work on time
Form a testing framework to simplify the debugging process
Create a fully-functional, bug-free application
METHODOLOGY
We will be dividing the team into 3-4 subgroups, including some or all of the following: Research/Documentation, Android/Java Implementation, Server/Web, and GUI & Graphics. In

addition, we will divide the development process into several discrete stages. The first stage will be dedicated to getting everyone in the group acclimated to the new development platform. We will do this by finding and sharing resources (documentation, sample code, etc.), as well as running through the set-up process for the Android SDK as a group. Before beginning development, we will establish documentation guidelines and testing frameworks in order to enhance maintainability and prevent bugs in the long run.
Next, we will work up specifications, establish deadlines, define roles, and allocate tasks to each group member. We plan on dividing up the development process into two central phases, core development and feature development. Core development will involve implementing the essential aspects of our application (calculator, transaction history, auto-sync, etc.), whereas feature development will encompass any additions which extend the app’s capabilities (PayPal, advanced GUI, e-mail/print functions, etc.). We will implement these additional features in order of priority (which we will establish before beginning development).
HISTORY
There are several applications “on the market” which are similar to our proposed app (examples include Bills Are In, Share a Bill, Split a Bill, Fair Share, and Xpense Split). A majority of these are web-based applications with little or no mobile functionality. In addition, they all require users to set up an account before they can utilize their services. The remaining apps we found were limited to the iOS platform, leaving the Android market open. Furthermore, we plan on adding some features which are not currently available in any of these applications.
46